HICUM/L2 version 2.33

Release Notes

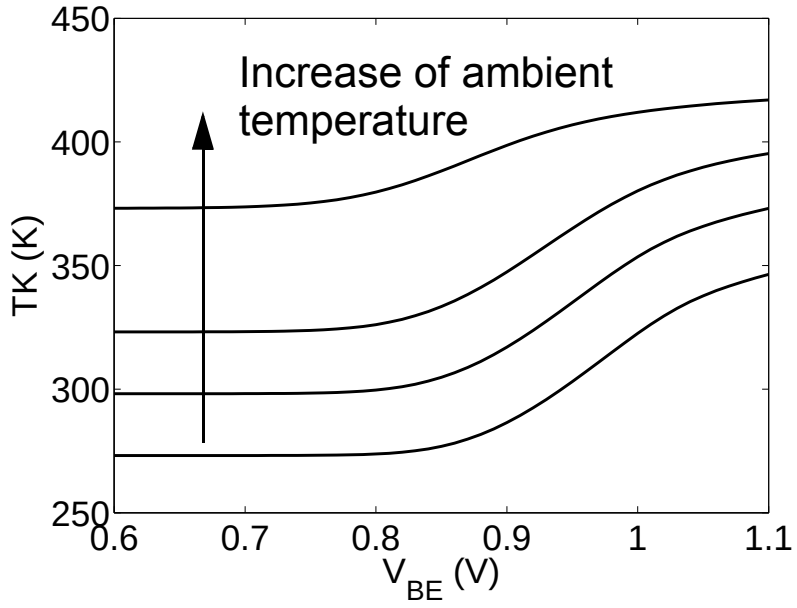
CEDIC, December 2013

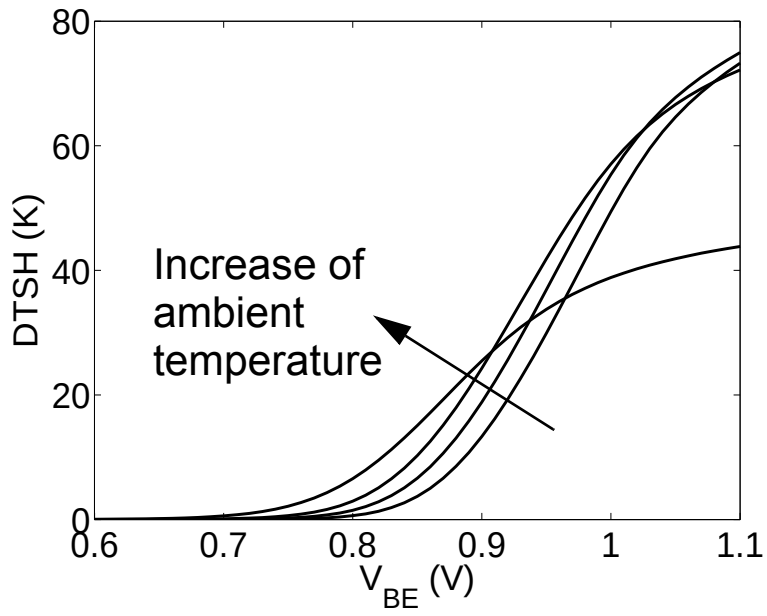
michael.schroeter@tu-dresden.de
andreas.pawlak@tu-dresden.de
anindya.mukherjee@tu-dresden.de

Operating point values

Additional operating point values (as per user request)

- TK: actual device temperature in Kelvin
- DTSH: temperature increase due to self-heating





Implementation flags

- Calculation of operating point values during transient simulation was requested
  - can cause significant increase of simulation time during transient simulations due to derivatives being calculated at each time point and iteration
  - Defining the most comprehensive description of the model
- Code implementation
  - Officially released code: OP-values are always calculated
  - Only for internal usage: two compiler flags are available (default in parentheses)
    - CALC_OP (defined) -> Operating point section is turned on (undefine to decrease simulation time)
    - OP_STATIC (undefined) -> Operating point values only calculated during ***DC*** simulation
    - Code:

```
`ifdef CALC_OP
    `ifdef OP_STATIC
        if (analysis("static")) begin: OPERATING_POINT
    `else
        begin: OPERATING_POINT
    `endif
```

→ Suggestion from model developer:

  - Named block in VA for OP-values -> no derivatives of values calculated there
  - Optimized implementation of model into simulators (no loss of simulation speed)

Operating point values

Bug-fixes

- Possible division by zero for DC and AC *current gain* was removed by conditional statement and addition of g_{min}
  - Code changes

```
IB    = I(<b>);
IC    = I(<c>);
...
if (IB != 0) begin
    BETADC = IC/IB;
end else begin
    BETADC = 0;
end
...
BETAAC = GMi/(gPIi+gPIx+`Gmin);
```

- Moved OP-value block out of noise block

Bug-fixes

- Moved voltage accesses below Model_Initialization block
  - Code changes

```
Vbiei = type*V(br_biei);
....
`MODEL begin : Model_initialization
```

was changed to

```
end     //of Model_initialization

Vbiei = type*V(br_biei);

...
```

- Additional usage of abs() in white_noise sources
  - Diode currents
  - Avoid issues for reversed biased pn-junctions

```
I(br_bpei)      <+ white_noise(twoq*abs(ibep),  "ibep");
```

Ranges of NQS parameters

- Allowd ranges of alit and alqf was reset
  - Allows 0 for both again
  - Improved conditional statements of NQS and correlated noise block
    - At least one of both parameters need to be larger 0
  - Also added correct backward compatibility
  - Code changes:

Parameter definition:

```
parameter real alqf     = 0.167          from [0:1];
parameter real alit     = 0.333          from [0:1];
```

Conditional statements:

```
if ((flnqs != 0 || flcomp == 0.0 || flcomp == 2.1) &&
     Tf != 0 && (alit > 0 || alqf > 0)) begin
```

and

```
if ( flcono==1 && (alit > 0 || alqf > 0)) begin
```

Noise sources

- Added descriptive names to noise sources
  - Code changes:

```
I(br_bbp_i)      <+ white_noise(fourkt/rbx_t,     "thermal");
```

  was changed to

```
I(br_bbp_i)      <+ white_noise(fourkt/rbx_t,     "rbx");
```

  - Done for all noise sources
- Added abs() for all white noise sources

Code optimization

- Added ibets>0 to conditional statements for all tunneling current blocks
  - Reduces executed code for tunneling current turned off
  - Code changes:

```
if (ibets > 0) begin : HICTUN_T
```

  - Done for all blocks relating the tunneling current
- Simplification of current gain formulation for correlated noise block

```
betad=ibei;
if (betad ibei > 0.0) begin
  betadin=betad;
  betan=it;
  betadc=betan/betad it/ibei;
end else begin
  betadc=0.0;
end
```